Project Proposal for CSE 517

David Simmons, Bohan Song, Moses Prasad Varghese
University of Washington
{simmons3, bs801, mosespv}@uw.edu

Citation and URL

Our proposed project is to reproduce the experiments in the paper *Learning Implicit Sentiment in Aspect-based Sentiment Analysis with Supervised Contrastive Pre-Training* by Li, Zou, Zhang, Zhang, & Wei (1) which can be found at the following URL https://aclanthology.org/2021.emnlp-main.22/.

Main Hypothesis

Aspect-level sentiment analysis traditionally focus on explicit sentiment representation, rather than implicit sentiment. In this paper, the researchers use a method called Supervised ContrAstive Pre-Training Learning (SCAPT) to augment the training data and capture more of the structure of implicit sentiment. In addition, classification will be proceeded with two other famous NLP models, Transformer Encoder and BERT model. In the end, each performance of two fine-tuned model together with SCAPT compares to 14 models that other researchers have constructed on the same public data. It shows that on average adding SCAPT roughly increases accuracy by 5% over the base models and accuracy of predicting implicit sentiment expression (ISE) by 10% over the base models. The main result that we try to reproduce is shown in the figure below (taken from Figure 1 in (1).

Method		Restaurant				Laptop			
		Acc.	F1	ESE	ISE	Acc.	F1	ESE	ISE
Ours	TransEncAsp	77.10	57.92	86.97	48.96	65.83	59.53	74.31	43.20
	BERTAsp	85.80	78.95	92.73	63.67	78.53	74.07	82.33	68.39
	BERTAsp+CEPT	87.50	82.07	93.67	67.79	81.66	78.38	83.84	75.86
	TransEncAsp+SCAPT	83.39	74.53	88.04	68.55	77.17	73.23	78.70	72.82
	BERTAsp+SCAPT	**89.11**	**83.79**	**94.37**	**72.28**	**82.76**	**79.15**	**84.70**	**77.59**

Access to the Data

The sentiment analysis is conducted mainly on two benchmarks, restaurant and laptop reviews https://paperswithcode.com/dataset/semeval-2014-task-4-sub-task-2. We combine the above data-set with a more challenging Multi Aspect Multi-Sentiment (MAMS) based data https://paperswithcode.com/dataset/mams to make our model more robust. We retrieve sentiment-annotated corpora from document level labeled data for pre-training by extracting five-stars-rated/one-star-rated reviews from the Yelp and Amazon Review https://paperswithcode.com/dataset/amazon-product-data and https://www.yelp.com/dataset.

Code Repository

The paper has a GitHub repository https://github.com/tribleave/scapt-absa. Since the code is quite lengthy, we intend to do a combination of implementing our own code and reusing some of their code. More specifically, we are interested in implementing the model portion of the code that involves SCAPT, the transformer encoder, and BERT. These files are found in the /model and /train sub-directories. The pre-processing portion of the code we will reuse.

Discussion of Feasibility

While there a few challenges to the project, we believe it be doable. The code we intend to reproduce in the paper's implementation consists of about 2000 lines. Across the three of us, we will aim to get the code for the base models TransEncAsp and BERTAsp completed by week 6 of the quarter. Once that is working, we can implement SCAPT from week 7-9 giving us a week to write the final report. The scale of the retrieved sentiment corpora from Yelp and Amazon reviews used in SCAPT contains millions of sentences of data. The scale of this dataset will likely provide some challenges, depending on our computational resources, and force us to reduce the training size. Even so we should be able to demonstrate that SCAPT improves the base models performance on implicit sentiment classification.

References

[1] Li, Z., Zou, Y., Zhang, C., Zhang, Q. & Wei, Z. Learning Implicit Sentiment in Aspect-based Sentiment Analysis with Supervised Contrastive Pre-Training. *Proceedings Of The 2021 Conference On Empirical Methods In Natural Language Processing*. pp. 246-256 (2021,11), https://aclanthology.org/2021.emnlp-main.22

CSE 517 Project Report - SCAPT

David Simmons, Bohan Song, Moses Prasad Varghese
University of Washington
{simmons3, bs801, mosespv}@uw.edu

Reproducibility Summary

Scope of Reproducibility

The paper “Learning Implicit Sentiment in Aspect-based Sentiment Analysis with Supervised Contrastive Pre-Training" presents a pre-training model for implicit sentiment classification. The authors claim that this pre-training improves classification performance, especially on the implicit aspect-based sentiment classification task.

Methodology

To provide evidenence to support the claim of the paper, we compared two language models TransEncAsp and BERTAsp trained on Laptop and Restaurant reviews with and without SCAPT to show that pre-training helps the model learn implicit sentiment expression. We use the code from the original paper’s GitHub repository. Without the pre-training of SCAPT, the models were fairly quick to train, taking only a matter of a few minutes. We are still in the processing of testing the code for the pre-trainining, but an initial run showed that training a single epoch took upwards of 75-120 minutes for each model.

Results

In the end, we were able to only partially reproduce the results in “Learning Implicit Sentiment in Aspect-based Sentiment Analysis with Supervised Contrastive Pre-Training" as computational resources were a more of a limiting factor than initially expected. We had to reduce batch size and the size of the external corpora in order to successfully train the models with SCAPT in a timely manner. This led to only slight improvement of BERTAsp+SCAPT over BERTAsp and no clear improvement of TransEncAsp+SCAPT over TransEncAsp.

What was Easy

Training without SCAPT on a few thousand review corpus takes least amount of time and resource. Since author has a fully detailed description of key commands on their Github page, it is easy to comprehend the entire code running procedure. We are able to easily understand the input data format (XML file and its associated pkl file) and produce a new input data from Best buy.

What was Difficult

Selecting batch size, learning rate and size of training data is really difficult for us. Every choice of hyperparameter takes hours to train and find out which ones yields a better outcome. While training on Colab, we have to manually keep the session active.

Communication with Original Authors

One of the authors, Zhengyan Li, responded to our email. He answered our questions about notation in the paper and commented on SCAPT pretraining and fine-tuning graph. With this feedback, we were able to correct the diagram in our model description section.

1 Introduction

Our project is focused on the paper "Learning Implicit Sentiment in Aspect-based Sentiment Analysis with Supervised Contrastive Pre-Training" by Li, Zou, Zhang, Zhang, and Wei (1). There the authors present a new approach to classifying aspect-based review sentiment. They note that many reviews ($\sim$ 27-30%) do not contain explicit opinion words but still convey implicit sentiment. Many models perform quite poorly for these reviews which they attribute to the challenges of small training datasets (only thousands of sentences). These datasets are insufficient to learn good sentiment embeddings. To overcome this obstacle, they propose and implement a pre-training model, Supervised ContrAstive Pre-Training (SCAPT), which leverages external corpora of datasets to learn aspect and sentiment embeddings. These embeddings significantly improve language models' sentiment classification accuracy and F1 score. However, while there is a moderate improvement on sentences with explicit sentiment expression (ESE), the significant improvement they see is when dealing with implicit sentiment expression (ISE), supporting their claim for having "learned implicit sentiment."

2 Scope of Reproducibility

We decide to reproduce the main experiment of the paper to support their claim that Supervised ContrAstive Pre-Training (SCAPT) significantly improves language models' implicit sentiment classification accuracy. This consists of training and testing the two models, TransEncAsp and BERTAsp, the first based on the transformer encoder and the second on BERT. There are datasets of Laptop and Restaurant reviews on which they separately train and evaluate. We will train and test these models without pre-training and report the accuracy and F1 scores for the whole dataset as well as the ESE and ISE slices as a baseline. Then we will apply the pre-training of SCAPT and fine-tune TransEncAsp and BERTAsp. These models with the pre-training are called TransEncAsp+SCAPT and BERTAsp+SCAPT. In particular, TransEncAsp+SCAPT and BERTAsp+SCAPT achieve a significant improvement of accuracy on implicit sentiment detection when compared to the models without SCAPT. Moreover, TransEncAsp+SCAPT even outperforms BERTAsp. This is quite impressive because of the fact that BERT has been pretrained on massive amounts of data with massive amounts of computation hours whereas TransEncAsp+SCAPT has significantly cheaper pre-training.

2.1 Addressed Claims from the Original Paper

The two main claims we are testing are the following:

1. SCAPT helps improve overall model accuracy in sentiment classification tasks where we can leverage external corpora of data.
2. SCAPT drastically improves language model's accuracy on reviews which only contain implicit sentiment expression (ISE) which demonstrates it's an effective method for learning implicit sentiment.

To support these claims, we will reproduce the performance results of the following table for the models TransEncAsp, BERTAsp, TransEncAsp+SCAPT, BERTAsp+SCAPT on Laptop and Restaurant reviews:

Method		Restaurant				Laptop			
		Acc.	F1	ESE	ISE	Acc.	F1	ESE	ISE
Ours	TransEncAsp	77.10	57.92	86.97	48.96	65.83	59.53	74.31	43.20
	BERTAsp	85.80	78.95	92.73	63.67	78.53	74.07	82.33	68.39
	BERTAsp+CEPT	87.50	82.07	93.67	67.79	81.66	78.38	83.84	75.86
	TransEncAsp+SCAPT	83.39	74.53	88.04	68.55	77.17	73.23	78.70	72.82
	BERTAsp+SCAPT	**89.11**	**83.79**	**94.37**	**72.28**	**82.76**	**79.15**	**84.70**	**77.59**

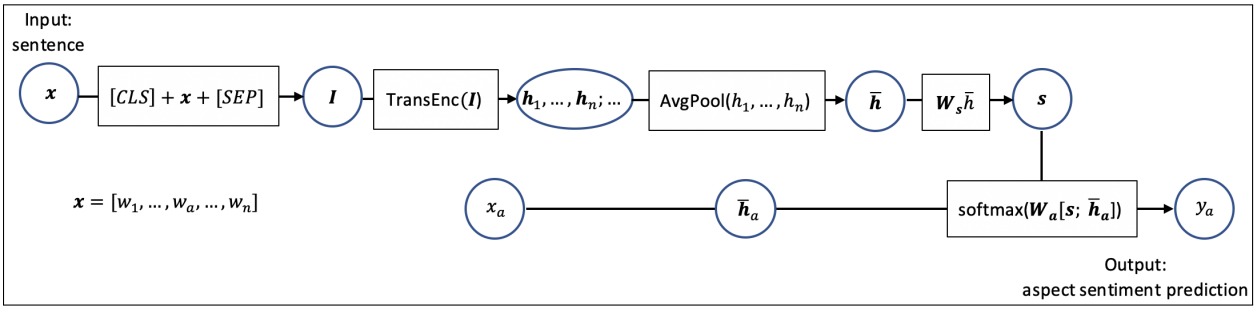
Table 1: From Table 3 in (1)

3 Methodology

3.1 Model Descriptions

In (1), they build two main models, TransEncAsp and BERTAsp, the first having the transformer encoder at its core and the second having BERT. For concreteness, we will focus on TransEncAsp. BERTAsp has the exact same architecture with the transformer encoder replace with BERT. Given a sentence $\mathbf{x} = [w_1, \ldots, w_a, \ldots, w_n]$ with aspect x_a at token

w_a, we predict the aspect sentiment y_a. This is a probability vector of positive, neutral, and negative sentiment for the aspect x_a in sentence $\mathbf{x}$. The first step is to reformat the sentence by adding a start token [CLS] and end token [SEP]: $I = [CLS] + \mathbf{x} + [SEP]$. The vocabulary indexed version of this is passed into the transformer encoder which computes hidden representations $\mathbf{h}_1, \ldots, \mathbf{h}_n$ of each token. These are the passing through an average pooling layer which gives the sentence representation $\bar{\mathbf{h}}$. Applying a learnable matrix transformation $\mathbf{W}_s$ to this gives the sentiment embedding s. The second piece of the input is the aspect x_a in $\mathbf{x}$. All of the hidden representations of occurrences of the aspect x_a in the train dataset are pooled together to form an aspect embedding $\bar{\mathbf{h}}_a = \text{AvgPool}(h_a)$. The final layer of TransEncAsp is to concatenate s and $\bar{\mathbf{h}}_a$, apply another learnable matrix transformation $\mathbf{W}_a$, and then take the softmax to obtain y_a.



TransEncAsp Architecture Diagram

For TransEncAsp, they randomly initialized parameters for the transformer encoder and the linear layers. For BERTAsp, they initialize with the pre-trained BERT parameters. The baseline performance without SCAPT is obtained by training these on the Laptop and Restaurant train datasets with the cross-entropy loss $\mathcal{L} = -\sum_{\mathbf{x}\in\mathcal{D}} \log y_a$.

Supervised ContrAstive Pre-Training (SCAPT) is the approach to improve learn better aspect and sentiment embeddings with which to initial the parameters in TransEncAsp and BERTAsp, before fine-tuning using the Laptop and Restaurant train datasets with the cross-entropy loss as above. When doing this, we will call the models TransEncAsp+SCAPT and BERTAsp+SCAPT.

SCAPT is split into three objectives: 1) contrastive sentiment embeddings (this has the parameters which want to take from the pretraining and use in), 2) review reconstruction, and 3) masked aspect prediction.

1) Contrastive sentiment embeddings is described above for TransEncAsp but has associated probability and loss for a batch B given by

$$P_B^{sup}(i,c) = \frac{\exp(s_i \cdot s_c/\tau)}{\sum_{b\in B, b\neq i} \exp(s_i \cdot s_b/\tau)}, \qquad \mathcal{L}_B^{sup} = \sum_{i\in B} -\log \frac{1}{C_i} \sum_{y_i=c, c\neq i} P_B^{sup}(i,c)$$

where C_i is the number of sentences in B with the same sentiment as $\mathbf{x}_i$. This loss pulls embeddings with the same sentiment closer together and pushes ones with different sentiment apart.

2) Review reconstruction passes the hidden embedding through a transformer decoder to reconstruct the original sentence and has probability and loss given by

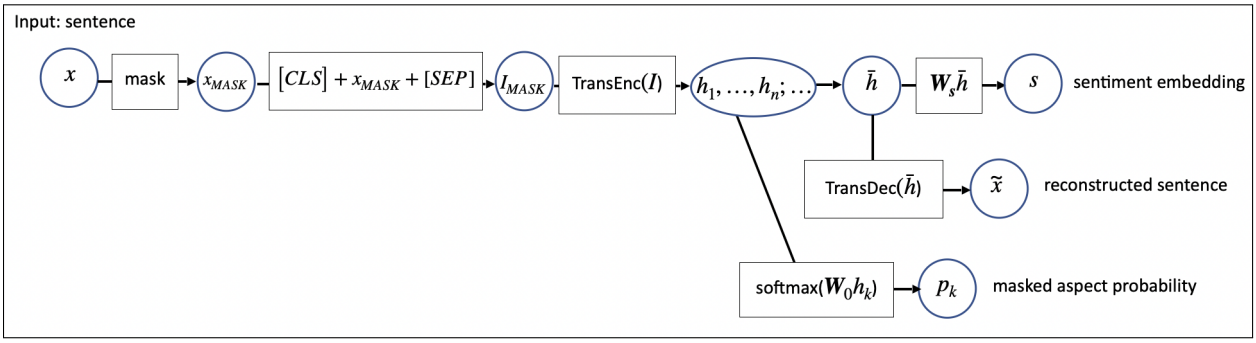
$$P^{rec}(\tilde{\mathbf{x}}) = \text{TransDec}(\bar{h}), \qquad \mathcal{L}_B^{rec} = -\sum_{\mathbf{x}\in B} \log P^{rec}(\tilde{\mathbf{x}}).$$

This helps the hidden embeddings retain information about their original sentence.

3) Masked aspect prediction randomly masks aspect tokens and other tokens to keep the embeddings from picking up too much noise from the pre-training data. It takes the hidden representations of the masked positions k and tries to predict the masked tokens. It has probability and loss given by

$$P^{map}(k) = \text{softmax}(\mathbf{W}_o h_k), \qquad \mathcal{L}_B^{map} = \sum_{\mathbf{x}\in B, x_k = w_{MASK}} -\log P^{map}(k).$$

Below is a diagram which illustrates how these three steps combine to form SCAPT.



SCAPT Architecture Diagram

For pre-training, we use a combination of the losses above

$$\mathcal{L}_B^{pre} = \mathcal{L}_B^{sup} + \alpha\mathcal{L}_B^{rec} + \beta\mathcal{L}_B^{map}$$

where α and β are hyperparameters selected to balance the weight of each loss.

3.2 Datasets

Our main benchmark of dataset are the Laptop and Restaurant reviews which are obtained from the SemEval 2014 task 4 `https://paperswithcode.com/dataset/semeval-2014-task-4-sub-task-2`. All these datasets involve three sentiment categories which are positive, neutral, and negative. We have 4722 restaurant reviews out of which we take 24 percent as test set and similarly we have 2951 laptop reviews out of which we take 22 percent as test set. All the data set used for developing the model are in xml format. For each data set we have the sentence ID, sentence text, aspect terms for positive, neutral and negative opinion and it respective polarity. For implicit data set we specify the opinion word and also if implicit sentiment is present or not. These classification is already done on the data set which we use for training the model and further the test data. For SCAPT, we pretrain the model using the external corpora of reviews from Amazon and Yelp found at `https://paperswithcode.com/dataset/amazon-product-data` and `https://www.yelp.com/dataset`. Concretely, we use Yelp data when dealing with Restaurant reviews, and Amazon for Laptop reviews. We have about 1.56/0.51 million sentence-level reviews from Yelp/Amazon. One example from the dataset of laptop reviews is as follows: Sentence id “2339":

I charge it at night and skip taking the cord with me because of the good battery life.

The aspect term “cord" has polarity “neutral" but implicit sentiment and the aspect term “battery life" has polarity “positive" which is explicit sentiment since it has the opinion word “good."

3.3 Hyperparameters

Our hyperparameters for the SCAPT model are α, β and τ which are the coefficients of the review reconstruction loss, masked aspect prediction loss and the temperature for supervised contrastive learning, respectively. We set the parameters, α and β to 1 and τ to 0.07 based on the analysis from the paper and the code. We apply SCAPT to TransEncAsp and BERTAsp, and these models are fine-tuned by a method called aspect-aware fine-tuning which is based on the opinion words, polarity and implicit sentiment. The framework for the TransEncAsp model is a 300-dimensional randomly initialized transformer encoder with 6 layers and 6 heads and for BERTAsp we have the BERT-base-uncased model as the basis. When we apply the laptop and restaurant review dataset directly to the TransEncAsp and BERTAsp models for training, we set the number of epochs to 80 and 8 respectively and we use the Adam optimizer with learning rate of 10^{-3} for TransEncAsp and $5 * 10^{-5}$ for BERTAsp. As mentioned before we make the training more robust by the aspect-aware fine-tuning and for this training we use a learning rate of $5 * 10^{-5}$. Here are some of the more details:

For BERTAsp model on laptop review dataset: batch size: 32, epoch: 6, optimizer: adamw, learning rate: 0.00005

For BERTAsp model on restaurant review dataset: batch size: 32, epoch: 8, optimizer: adamw, learning rate: 0.00005

For TransEncAsp model on laptop review dataset: batch size: 32, epoch: 10, head size: 6, layers: 6, hidden size: 300, feedforward: 1200, dense hidden size: 100, optimizer: adamw, learning rate: 0.0001

For TransEncAsp model on restaurant review dataset: batch size: 32, epoch: 20, head size: 6, layers: 6, hidden size: 300, feedforward: 1200, dense hidden size: 100, optimizer: adamw, learning rate: 0.0001

For BERTAsp+SCAPT model on amazon review dataset for pre-training: batch size: 16, epoch: 8, decoder heads: 6, decoder layers: 6, decoder hidden: 768, optimizer: adamw, lambda scl: 2.0 (Tau), lambda map: 1.0 (beta), lambda rr: 1.0 (alpha), learning rate: 0.00005, learning rate bert: 0.00005

For TransEncAsp+SCAPT model on amazon review dataset for pre-training: batch size: 16, epoch: 80, head size: 6, layers: 6, hidden size: 300, feedforward: 1200, dense hidden size: 100, decoder heads: 6, decoder layers: 6, decoder hidden: 300, optimizer: adamw, lambda scl: 2.0 (Tau), lambda map: 1.0 (beta), lambda rr: 1.0 (alpha), learning rate: 0.001, learning rate bert: 0.001

For BERTAsp+SCAPT model on Yelp review dataset for pre-training: batch size: 16, epoch: 8, decoder layers: 6, decoder hidden: 768, decoder heads: 8, optimizer: adamw, lambda scl: 2.0 (Tau), lambda map: 1.0 (beta), lambda rr: 1.0 (alpha), learning rate: 0.00005, learning rate bert: 0.00005

For TransEncAsp+SCAPT model on Yelp review dataset for pre-training: batch size: 16, epoch: 80, head size: 6, layers: 6, hidden size: 300, feedforward: 1200, dense hidden size: 100, decoder heads: 6, decoder layers: 6, decoder hidden: 300, optimizer: adamw, lambda scl: 2.0 (Tau), lambda map: 1.0 (beta), lambda rr: 1.0 (alpha), learning rate: 0.001, learning rate bert: 0.001

3.4 Implementation

We are planning to use the existing code for the project. The link to our fork of their GitHub repository is `https://github.com/davidsimmons314/SCAPT-ABSA`. The primary programming language used for the model is Python. Some of the major packages used for implementing the model are PyTorch, transformers for BERT and SCAPT among many others.

3.5 Experimental Setup

We fork their repository and set up in Google Colab.The spec of CPU is 2-Core Intel(R) Xeon(R) and the spec of GPU is NVIDIA-SMI. The requirments are listed below.

- cuda 11.0
- python 3.7.9
- lxml 4.6.2
- numpy 1.19.2
- pytorch 1.8.0
- pyyaml 5.3.1
- tqdm 4.55.0
- transformers 4.2.2

3.6 Computational Requirements

There are a total of 8 conducted experiments, where we train TransEncAsp, TransEncAsp+SCAPT, BERTAsp, and BERTAsp+SCAPT on laptops (Amazon) reviews or restaurants (Yelp) reviews separately. Computation requirement only differentiates when the model adopt the SCAPT preprocessing.

— Preprocessing

SCAPT is computationally expensive. More specifically, SCAPT will take roughly 55 minutes to train 1 epoch on Amazon laptop reviews (0.38M), whereas it will take 150 minutes to train an epoch on Yelp restaurant reviews(1.17M). In the original paper, they trained 80 epochs. Since we do not have access to super powerful GPU, we decide to only train on 10 epochs with larger learning rate. Before we conduct the experiment, we thought the computation would be much less because numerous of reviews is only composed of a sentence, sometimes a few words.

—Training and Evaluation

When it comes to training both models, computation appears to the same in a sense that one epoch on Amazon review is about 5 seconds and one epoch on Yelp is about 8-12 seconds. The evaluation do not require any computation power. This is exactly what we expected before we conduct our experiment.

4 Results

We were able to partially reproduce the results as computational resources were a more of a limiting factor than initially expected. We had to reduce batch size and the size of the external corpora in order to successfully train the models with SCAPT in a timely manner. BERTAsp+SCAPT on laptops gained 2.86% mroe ISE accuracy than BERTAsp while BERTAsp+SCAPT on restaurant gained 0.75% more than BERTAsp. On the other hand, TransEncAsp+SCAPT on laptops gained 2.29% more ISE accuracy thanTransEncAsp , but TransEncAsp+SCAPT on restaurant gives a worse ISE accuracy. Thus we had slight improvement of BERTAsp+SCAPT over BERTAsp and no clear improvement of TransEncAsp+SCAPT over TransEncAsp.

Below is the table for the results that we obtained when training and evaluating the authors code.

Model	Test Set	Accuracy	F1 Score	ESE	ISE
TransEncAsp	Laptops	62.85	55.87	69.33	45.71
TransEncAsp	Restaurant	72.86	60.47	80.66	47.94
BERTAsp	Laptops	78.21	73.55	83.15	65.14
BERTAsp	Restaurant	85.36	78.84	91.56	65.54
TransEncAsp+SCAPT	Laptops	63.17	56.22	68.90	48.00
TransEncAsp+SCAPT(250k)	Restaurant	72.32	59.24	78.66	45.32
BERTAsp+SCAPT(250k)	Laptops	77.74	73.53	81.43	68.00
BERTAsp+SCAPT(250k)	Restaurant	85.89	79.05	92.03	66.29

The first four rows of this table show the results of the full training on the base models TransEncAsp and BERTAsp. We were able to run the training without changing any hyper parameters and achieved results similar to those reported in Table 3 of (1) within a reasonably small variance. However, for the models where we first pre-trained with SCAPT we were not able to fully replicate their results. Due to computational constraints (see discussion below), we had to reduce the batch size, number of training epochs, and size of the external corpora datasets: to 250k from Amazon, down from $\sim$ 500k, and to 250k from Yelp, down from $\sim$ 1500k.

For BERTAsp+SCAPT, we were able to do the pre-training for the same number of 8 epochs as in the paper but had to reduce the batch size from 16 to 8 as we ran out of GPU memory otherwise. The ISE of BERTAsp+SCAPT on both the Laptops and Restaurant datasets did slightly improve, but it is not clear that is improvement is statistically significant.

For TransEncAsp+SCAPT, we had to dramatically reduce the number of training epochs since we could not do near the full 80 epochs (this is much higher than for BERT since TransEncAsp is randomly initialized whereas BERT is initialized with the available pretrained values). We could only do 9/80 on the Laptops data (this was with the full Amazon dataset), and 38/80 on the Restaurant dataset (this was with 250k YELP data) and also had to reduce batch size.

4.1 Additional Datasets

To further test our models with and without SCAPT, we created an additional dataset of laptop reviews from Best Buy. Our dataset can be found at the following link: `https://github.com/davidsimmons314/SCAPT-ABSA/blob/master/data/laptops/bestbuy_laptops.xml`. We created this dataset using reviews from the two links : `https://www.bestbuy.com/site/dell-inspiron-7000-2-in-1-14-touch-screen-laptop-amd-ryzen-5-8gb-memory-256gb-solid-state-drive-blue/6458907.p?skuId=6458907` and `https://www.bestbuy.com/site/combo/macbook-air/a0884979-2062-4269-a9c4-9f416af8cb31#ratingsAndReviewsAccordion`. From these online Best Buy customer reviews we gathered 51 unique text reviews of laptops of two brands, Dell and Apple to evaluate on the models. These reviews as texts were formatted in XML. We searched through the reviews to select the appropriate ones by focusing on different features like size, weight, screen, RAM, display, battery life, charging etc and gather ample amount of data which shows neutral, positive and negative polarity. We gathered 20 implicit sentences and 31 explicit sentences. We have 43 unique aspect terms and 41 positive polarity texts, 27 negative polarity texts and 4 neutral polarity texts. The total polarity texts exceeds 51 because some texts have multiple polarity with different aspect terms and opinion word. Each text was analysed separately and the opinion words were determined based on the context representing the aspect term and the meaning of the sentences. The data set already given to us as different parameters such as sentence id, text, aspect term, polarity, range, opinion words and condition to check if its implicit. We used these parameters to create our own data set. The sentence id is in

the form of ratio representing a 3 digit unique identifier for the text. The text data was inputted from the above two links without any changes to preserve the originality of the review and to get more practical results when we use it on unknown data. As mentioned above the aspect term was handpicked based on the context of the sentence and what it was mainly referring to. Some texts had multiple aspect terms and those were separately analysed. For example, if the previous sentence contained two occurrences of the aspect term “performance”, there would be two <aspectTerm ... /> elements, which would be identical if both occurrences had negative polarity. Based on each aspect term we decided on the polarity and checked if the polarity was conveyed implicitly and accordingly determined the opinion words. Some texts seems to show opinion words that tally with the polarity but we took the general understanding of the word and decided whether it can be considered as a opinion word or not. The range "from" to "to" denotes the index of the aspect term in the text indicating its start and end offset in the text that was specifically needed for learning and to confirm the presence of the aspect term in the sentence. This was calculated for each text by analysing the size of the text, the beginning and end of the index of the aspect term.

The follow table shows the results of above trained models evaluated on this Best Buy dataset:

Model	Test Set	Accuracy	F1 Score	ESE	ISE
TransEncAsp	BestBuy	70.15	60.68	72.55	62.50
BERTAsp	BestBuy	86.57	59.93	92.16	68.75
TransEncAsp+SCAPT	BestBuy	67.16	52.08	70.59	56.25
BERTAsp+SCAPT(250k)	BestBuy	88.24	60.75	92.31	75.00

As above, BERTAsp+SCAPT does slightly improve over BERTAsp while TransEncAsp+SCAPT actually does worse.

4.2 Hyperparameter Tuning

With full data and default batch size, we have implimented different learning rate 0.001, 0.003 and 0.005 on Transformer with SCAPT. Since we have limited computing resource, we pick the best learning rate based on the accuracy and loss after 2 epochs, which takes about 2 hours. Although noise might play a role in early training stage, learning rate 0.005 seem to outperform than others in terms of faster decreasing training loss. The training accuracy with learning rate 0.005 is same as training accuracy with learning rate 0.003. Hence, we choose 0.005 as our learning rate on Transformer with SCAPT.

For Transformer with SCAPT and BERT with SCAPT on 250k data, we also changed batch size. although sentiment embedding in SCPAT require large batch size for better performance, we change the batch size from 32 to 16. Even with smaller batch size, we still encounter GPU RAM running out issues where many batches are skipped during training. In the end, we have to reduce batch size to 8 so that we do not receive any errors.

5 Discussion

The easy part was training the TransEncAsp and BERTAsp models without the SCAPT pretraining. Once we cloned the original code correctly on Colab and figured out how to enter correct commands, training both model on Laptops could be completed within 3 minutes, whereas training both model on restaurant only took less than 10 minutes.

We had a lot of difficulty in attempting to train SCAPT on local machine. On Colab, long period of training will trigger Captcha that disrupt the training process if the session is inactive. Therefore, we have to sit in front of computer for 8-10 hours and manually click the browser every hour to make the session active. Then, an appropriate subset of original data that can be trained with full epochs in less than 10 hours is also one of challenging task. We have tried pretraining BERT with SCAPT on 50k, 100k and 250k and realize more ISE improvement from 250k sample. In the end, we finalize our training data to be the first 250k reviews in each laptops and restaurant data.

For other scholars who might be interested in reproducing the main claim of this paper, batch size and number of GPU cards are the crucial components for SCAPT pretraining. This is clear not only due to our lack of significant improvement after having been forced to reduce batch size, but also due to the nature of the loss functions in SCAPT. As showed in the methodologies section, the formulas for $\mathcal{L}_B^{sup}$ and $\mathcal{L}_B^{map}$ are highly dependent on the batch size. We recommend training SCAPT on at least 2 powerful GPU card so that the training should be done in less than 48 hours. With enough computing power, one should keep batch size 32 untouched and experiment with different hyperparameters such as learning rate and number of connected layers in Transformer and BERT.

Communication with Original Authors

Even though we had read the paper thoroughly and discussed within our group many times, there were still a few notations which we had trouble to understand. For example, in the section 3.2 "Aspect Aware Fine Tuning", we were not sure whether average pooling layer was calculated on the aspect term within a sentence or a batch or entire data set. We also drew a flow chart in version 1 and we were not sure if the SCPAT flow chart precisely reflected the pretraining process.

We reached out to the authors of the paper to figure out these aspects. One of the authors, Zhengyan Li, responded to our email and clarified these questions. In response to our first question, as he illustrated, "the aspect-based representation $\hat{h}_a$ is obtained by doing average pooling on the aspect terms in the sentence x". As he commented on our graphs, "I think you may be misunderstood how to get the sentiment representation. They are linear transformed from the CLS position of TransEnc/BERT...And average pooling is only used when acquiring aspect representation."

We were able to use this feedback to correct the figures in our model description section.

References

[1] Li, Z., Zou, Y., Zhang, C., Zhang, Q. & Wei, Z. Learning Implicit Sentiment in Aspect-based Sentiment Analysis with Supervised Contrastive Pre-Training. *Proceedings Of The 2021 Conference On Empirical Methods In Natural Language Processing*. pp. 246-256 (2021,11), https://aclanthology.org/2021.emnlp-main.22